**Class 02**

**# What is Container?**

Containers are a software package into a logical box with everything that the application needs to run. That includes the operating system, application code, runtime, system tools, system libraries, and etc. Docker containers are built off Docker images. Since images are read-only, Docker adds a read-write file system over the read-only file system of the image to create a container.

Containers are compared with virtual machines (VMs). VMs are the guest operating system such as Linux or Windows runs on top of a host operating system with virtualized access to the underlying hardware. Containers allow you to package your application together with libraries and other dependencies, providing isolated environments for running your software services.

Different Container Providers

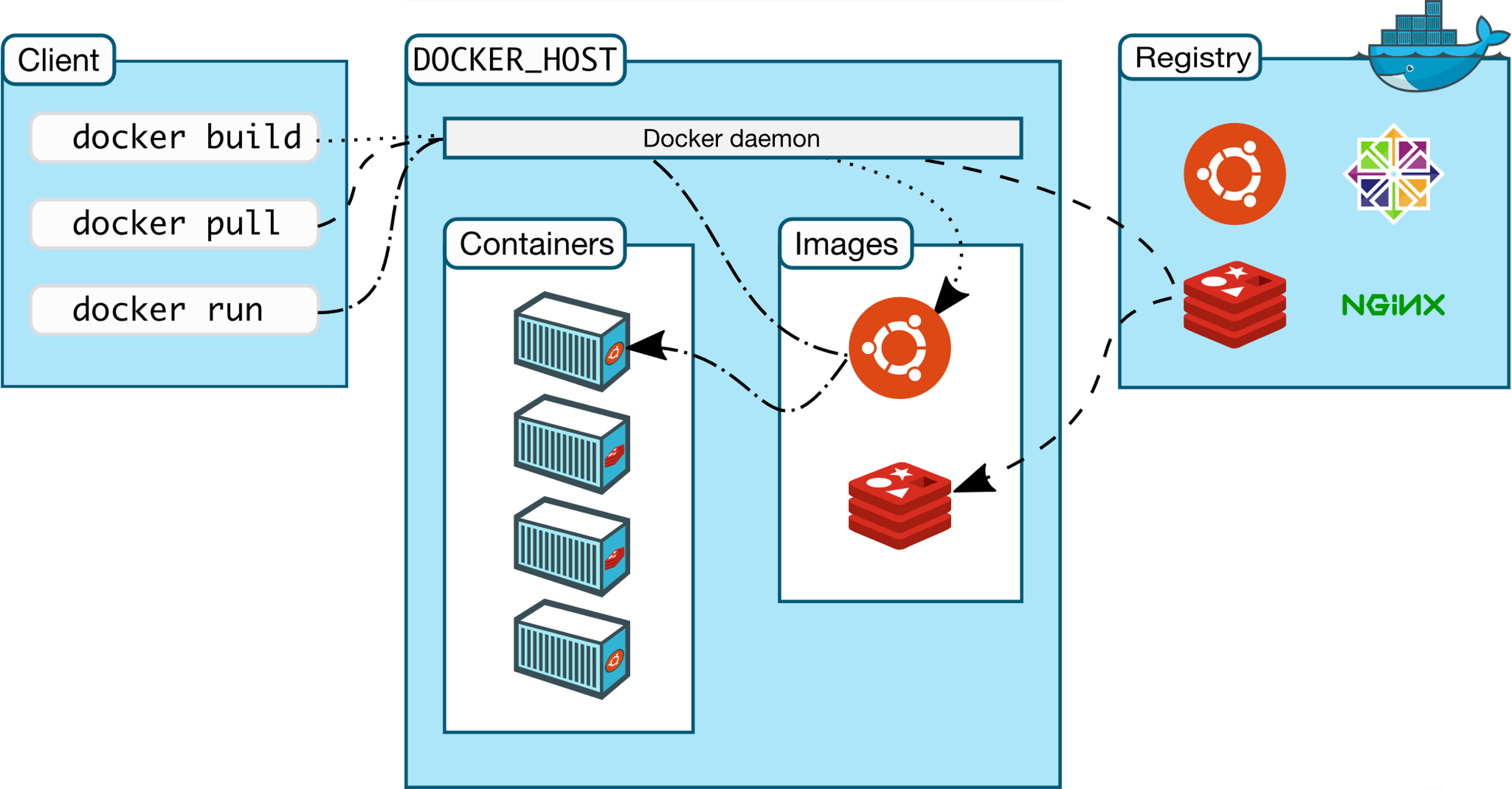
* Docker
* Microsoft Azure
* Amazon Web Services (AWS)
* Google Cloud Platform
* Linux Containers

**# What is Docker?**

Docker is an open-source platform based on Linux containers for developing and running applications inside containers. Docker is used to deploy many containers simultaneously on a given host. Containers are very fast and lightweight because they don’t need the extra load of a hypervisor as they run directly within the host machine’s kernel.

**Docker Architecture and Components**

Docker uses a client-server architecture. The docker client talks to the Docker daemon, which used to building, running, and distributing the Docker containers. The Docker client and daemon communicate using a REST API, over UNIX sockets, or a network interface.



There are five major components in the Docker architecture:

a) **Docker Daemon** listens to Docker API requests and manages Docker objects such as images, containers, networks and volumes.

b) **Docker Clients:** With the help of Docker Clients, users can interact with Docker. Docker client provides a command-line interface (CLI) that allows users to run, and stop application commands to a Docker daemon.

c) **Docker Host** provides a complete environment to execute and run applications. It comprises of the Docker daemon, Images, Containers, Networks, and Storage.

d) **Docker Registr**y stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to use images on Docker Hub by default. You can run your own registry on it.

e) **Docker Images** are read-only templates that you build from a set of instructions written in Dockerfile. Images define both what you want your packaged application and its dependencies to look like what processes to run when it’s launched.

**# Dockerfile**

Dockerfile is a simple text file that consists of instructions to build Docker images.

**# Docker build**

Using docker build users can create an automated build that executes several command-line instructions in succession.

**# Merkle Tree**

**# Alpine**

Alpine Linux is a Linux distribution built around musl libc and BusyBox. The image is only 5 MB in size and has access to a package repository that is much more complete than other BusyBox based images. This makes Alpine Linux a great image base for utilities and even production applications.

**# Fork**